

[illegible][illegible]

[illegible][illegible]

(2)	54	DECLARATIONS	
(3)	91	ACTION ROUTINES	
(4)	142	FALSINIT	
(5)	246	FALSINIT_RENAME	
(6)	293	FALSINIT_XABCHN	
(7)	312	SUPPORT ROUTINES	
(8)	371	FALSINIT_DATXAB,	FALSINIT_PROXAB
(8)	372	FALSINIT-SUMXAB,	FALSINIT-RDTXAB
(8)	373	FALSINIT-ALLXAB,	FALSINIT-KEYXAB

```
0000 1 .TITLE FALACTINI - STATE TABLE ACTION ROUTINES
0000 2 .IDENT 'V04-000'
0000 3
0000 4
0000 5 *****
0000 6
0000 7 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9 * ALL RIGHTS RESERVED.
0000 10
0000 11 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16 * TRANSFERRED.
0000 17
0000 18 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20 * CORPORATION.
0000 21
0000 22 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24
0000 25 *****
0000 26
0000 27
0000 28
0000 29 ++
0000 30 Facility: FAL (DECnet File Access Listener)
0000 31
0000 32 Abstract:
0000 33
0000 34 This module contains action routines called by the state table manager.
0000 35
0000 36 Environment: VAX/VMS, user mode
0000 37
0000 38 Author: James A. Krycka, Creation Date: 16-JUN-1977
0000 39
0000 40 Modified By:
0000 41
0000 42 V03-002 JAK0136 J A Krycka 07-MAR-1984
0000 43 Change FAL$T_NAMESPEC and FAL$K_NAMESPEC to FAL$T_FILESPEC2
0000 44 and FAL$K_FILESPEC2.
0000 45
0000 46 V03-001 KRM0066 K Malik 23-NOV-1982
0000 47 Add FAL$INIT_RENAME routine.
0000 48 Change FAL$T_EXPANDED, FAL$K_EXPANDED, FAL$T_RESULTANT and
0000 49 FAL$K_RESULTANT symbols to FAL$T_EXPAND, FAL$K_EXPAND,
0000 50 FAL$T_RESULT and FAL$K_RESULT.
0000 51
0000 52 --
```



```
0000 54      .SBTTL  DECLARATIONS
0000 55
0000 56 :
0000 57 : Include Files:
0000 58 :
0000 59
0000 60      $DAPPLGDEF      ; Define DAP prologue symbols
0000 61      $DAPHDRDEF     ; Define DAP message header
0000 62      $DAPCTLDEF     ; Define DAP Control message
0000 63      $FABDEF        ; Define File Access Block symbols
0000 64      $FALWRKDEF     ; Define FAL Work Area symbols
0000 65      $NAMDEF        ; Define Name Block symbols
0000 66      $RABDEF        ; Define Record Access Block sym**
0000 67      $XABDEF        ; Define symbols common to all XABs
0000 68      $XABALLDEF     ; Define Allocation XAB symbols
0000 69      $XABDATDEF     ; Define Date and Time XAB symbols
0000 70      $XABFHCDEF     ; Define File Header Char XAB symbols
0000 71      $XABKEYDEF     ; Define Key Definition XAB symbols
0000 72      $XABPRODEF     ; Define Protection XAB symbols
0000 73      $XABRDTDEF     ; Define Revision Date and Time symbols
0000 74      $XABSUMDEF     ; Define Summary XAB symbols
0000 75
0000 76 :
0000 77 : Macros:
0000 78 :
0000 79 :      None
0000 80 :
0000 81 : Equated Symbols:
0000 82 :
0000 83
0000 84      ASSUME  DAP$Q_DCODE_FLG EQ 0
0000 85      ASSUME  FAL$Q_FLG EQ 0
0000 86
0000 87 :
0000 88 : Own Storage:
0000 89 :
```



```
0000 91 .SBTTL ACTION ROUTINES
0000 92 .PSECT FAL$CODE NOSHR,EXE,RD,NOWRT,BYTE
0000 93
0000 94 :++
0000 95 : Functional Description:
0000 96 :
0000 97 : This module contains action routines invoked by the state table
0000 98 : manager (FAL$STATE).
0000 99 :
0000 100 : The input parameters and completion codes listed below are applicable
0000 101 : for all of these action routines. Note that an action routine may use
0000 102 : R0-R7 and AP without restoring them on exit. R0 on exit, however, must
0000 103 : represent a status code to indicate success/failure of the routine or
0000 104 : a true/false condition, as appropriate. This status code is used by
0000 105 : the state table manager to advance to the next state.
0000 106 :
0000 107 : Calling Sequence:
0000 108 :
0000 109 : BSBW FAL$INIT (only routine at present)
0000 110 :
0000 111 : Input Parameters:
0000 112 :
0000 113 : R8 Address of FAL work area
0000 114 : R9 Address of DAP control block
0000 115 : R10 Address of FAB
0000 116 : R11 Address of RAB
0000 117 :
0000 118 : Implicit Inputs:
0000 119 :
0000 120 : None
0000 121 :
0000 122 : Output Parameters:
0000 123 :
0000 124 : R0 Status code
0000 125 : R1-R7 Destroyed
0000 126 : AP Destroyed
0000 127 :
0000 128 : Implicit Outputs:
0000 129 :
0000 130 : None
0000 131 :
0000 132 : Completion Codes:
0000 133 :
0000 134 : R0 1 = success; 0 = failure
0000 135 :
0000 136 : Side Effects:
0000 137 :
0000 138 : None
0000 139 :
0000 140 :--
```



```
0000 142 .SBTTL FALSINIT
0000 143
0000 144 :++
0000 145 : This routine initializes the FAL work area in preparation for the next file
0000 146 : access request. This includes setting up the FAB, RAB, NAM, and FHCXAB control
0000 147 : blocks and linking them together.
0000 148 :--
0000 149
0000 150 FALSINIT:: : Entry point
68 00000702 8F CA 0000 151 BICL2 #<<FALSM_ATT_MSG>!-- : Clear flags
0007 152 <FALSM_WILD>!--
0007 153 <FALSM_FTM>!--
0007 154 <FALSM_BLK_IO>!--
0007 155 0>,(R8)
0007 156 MOVL #DAPSK_VALID_R2F,- : Define which DAP messages are valid
000D 157 DAPSL_MSG_MASK(R9) : to receive from partner
000F 158 MOVB #DAPSK_RAC_D,- : Set previous RAC value to the DAP
0011 159 FALSB_RAC(R8) : default RAC value
0014 160
0014 161 :
0014 162 : Initialize the FAB, RAB, NAM, and FHCXAB control blocks.
0014 163 :
0014 164
0014 165 ASSUME FALSK_FAB GE FABSC_BLN
0014 166 ASSUME FALSK_RAB GE RABSC_BLN
0014 167 ASSUME FALSK_NAM GE NAMSC_BLN
0014 168 ASSUME FALSK_FHCXAB GE XABSC_FHCLEN
0014 169
0014 170 ASSUME FALSL_FAB+FALSK_FAB+FALSK_RAB+FALSK_NAM EQ FALSL_FHCXAB
0014 171
0014 172 $ZERO_FILL- : Zero FAB, RAB, NAM, FHCXAB blocks
0014 173 DST=(R10)- : which must be contiguous in memory
0014 174 SIZE=#<FALSK_FAB+FALSK_RAB+FALSK_NAM+FALSK_FHCXAB>
001C 175
001C 176 ASSUME FABS_BID+1 EQ FABS_BLN
001C 177 ASSUME RABS_BID+1 EQ RABS_BLN
001C 178 ASSUME NAMS_BID+1 EQ NAMS_BLN
001C 179 ASSUME XABS_COD+1 EQ XABS_BLN
001C 180
001C 181 MOVW #<FABSC_BLN@8+FABSC_BID>,-
0020 182 FABS_BID(R10) : Insert FAB block ID and length
0021 183 MOVAB FALST_FILESPEC(R8),- : Store address of file specification
0025 184 FABS_FNA(R10) : string buffer in FAB
0027 185 $SETBIT #FABS_V_NAM,FABS_FOP(R10);Denote NAM block is present
002C 186 MOVW #<RABSC_BLN@8+RABSC_BID>,-
0030 187 RABS_BID(R11) : Insert RAB block ID and length
0031 188 MOVAB FALSL_NAM(R8),R7 : Get address of NAM block
0036 189 MOVW #<NAMSC_BLN@8+NAMSC_BID>,-
003A 190 NAMS_BID(R7) : Insert NAM block ID and length
003B 191 MOVAB FALST_EXPAND(R8),- : Store address of expanded string
003F 192 NAMS_ESA(R7) : buffer in NAM block
0041 193 MOVB #FALSK_EXPAND,- : Store size of expanded string
0044 194 NAMS_ESS(R7) : buffer in NAM block
0046 195 MOVAB FALST_RESULT(R8),- : Store address of resultant string
004A 196 NAMS_RSA(R7) : buffer in NAM block
004C 197 MOVB #FALSK_RESULT,- : Store size of resultant string
004F 198 NAMS_RSS(R7) : buffer in NAM block
```



```
08 A7 01 90 0051 199      MOVB  #NAMS$ PWD,NAMS$B NOP(R7); Do not mask out password in node spec
    2C1D 8F B0 0055 200      MOVW  #<XAB$C FHCLEN@8+XAB$C FHC>,-
    02F4 C8      0059 201      FALS$_FHCXAB+XAB$B_CODTR8); Insert FHCXAB ID and length
                   005C 202
                   005C 203
                   005C 204 : Link only the RAB, NAM, and FHCXAB to the FAB at this time.
                   005C 205 : The other XABs will be chained in as required when the DAP Access message
                   005C 206 : is processed.
                   005C 207
                   005C 208
3C AB 5A D0 005C 209      MOVL  R10,RAB$_FAB(R11) ; Store FAB pointer in RAB
    0294 C8 DE 0060 210      MOVAL FALS$_NAMTR8),- ; Store NAM pointer in FAB
    28 AA      0064 211      FABS$_NAM(R10)
    02F4 C8 DE 0066 212      MOVAL FALS$_FHCXAB(R8),- ; Store FHCXAB pointer in XAB chain
    24 AA      006A 213      FABS$_XAB(R10)
    02F8 C8 DE 006C 214      MOVAL FALS$_FHCXAB+XAB$_NXT(R8),-
    7C A8      0070 215      FALS$_CHAIN_NXT(R8) ; Save address of next chain pointer
                   0072 216
                   0072 217 : Initialize XAB related fields in the FAL work area.
                   0072 218
                   0072 219
                   0072 220
    72 A8 B4 0072 221      CLRW  FALS$_RECEIVED(R8) ; Zero mask of received XABs to chain
    74 A8 D4 0075 222      CLRL  FALS$_ALLXABINI(R8) ; Zero list of ALLXABs initialized
    78 A8 D4 0078 223      CLRL  FALS$_KEYXABINI(R8) ; Zero list of KEYXABs initialized
                   007B 224
                   007B 225 : Initialize volume and directory name descriptors for possible wildcard
                   007B 226 : processing.
                   007B 227
                   007B 228
                   007B 229
                   007B 230      ASSUME FALS$_VOLNAME+8 EQ FALS$_DIRNAME
                   007B 231
50 0080 C8 7E 007B 232      MOVAQ FALS$_VOLNAME(R8),R0 ; Get address of first descriptor
    80 80 D4 0080 233      CLRL  (R0)+ ; Initialize volume name descriptor
    80 1E00 C8 9E 0082 234      MOVAB FALS$_VOLNAME(R8),(R0)+ ;
    80 80 D4 0087 235      CLRL  (R0)+ ; Initialize directory name descriptor
    80 1F00 C8 9E 0089 236      MOVAB FALS$_DIRNAME(R8),(R0)+ ;
                   008E 237
                   008E 238 : Exit state with success if a valid Configuration message has been processed;
                   008E 239 : otherwise, exit state with failure.
                   008E 240
                   008E 241
                   008E 242
50 68 01 00 EF 008E 243      EXTZV #FALS$_CNF_MSG,#1,(R8),R0
    05 0093 244      RSB ; Exit state with status code in R0
```



```
0094 246 .SBTTL FALSINIT_RENAME
0094 247
0094 248 :++
0094 249 : This routine initializes the secondary FAB and NAM control blocks in the FAL
0094 250 : work area which are used by the rename operation.
0094 251 :--
0094 252
0094 253 FALSINIT_RENAME:: ; Entry point
0094 254
0094 255 ASSUME FAL$K_FAB2 GE FAB$C_BLN
0094 256 ASSUME FAL$K_NAM2 GE NAM$C_BLN
0094 257
0094 258 ASSUME FAL$L_FAB2+FAL$K_FAB2 EQ FAL$L_NAM2
0094 259
5A 0800 C8 DE 0094 260 MOVAL FAL$L_FAB2(R8),R10 ; Get address of FAB2 in R10
0099 261 $ZERO_FILL- ; Zero FAB2 and NAM2 blocks which must
0099 262 DST=(R10)- ; be contiguous in memory
0099 263 SIZE=#<FAL$K_FAB2+FAL$K_NAM2>
00A1 264
00A1 265 MOVW #<FAB$C_BLN@8+FAB$C_BID>,-
00A5 266 FAB$B_BID(R10) ; Insert FAB block ID and length
0900 C8 9E 00A6 267 MOVAB FAL$T_FILESPEC2(R8),- ; Store address of file specification
2C AA 00AA 268 FAB$L_FNA(R10) ; string buffer in FAB
00AC 269 $SETBIT #FAB$V_NAM,FAB$L_FOP(R10); Denote NAM block is present
57 0850 C8 9E 00B1 270 MOVAB FAL$L_NAM2(R8),R7 ; Get address of NAM2 block
6002 8F B0 00B6 271 MOVW #<NAM$C_BLN@8+NAM$C_BID>,-
00BA 272 NAM$B_BID(R7) ; Insert NAM block ID and length
0A00 C8 9E 00BB 273 MOVAB FAL$T_EXPAND2(R8),- ; Store address of expanded string
0C A7 00BF 274 NAM$L_ESA(R7) ; buffer in NAM block
FF 8F 90 00C1 275 MOVAB #FAL$R_EXPAND2,- ; Store size of expanded string
0A A7 00C4 276 NAM$B_ESS(R7) ; buffer in NAM block
0B00 C8 9E 00C6 277 MOVAB FAL$T_RESULT2(R8),- ; Store address of resultant string
04 A7 00CA 278 NAM$L_RSA(R7) ; buffer in NAM block
FF 8F 90 00CC 279 MOVAB #FAL$R_RESULT2,- ; Store size of resultant string
02 A7 00CF 280 NAM$B_RSS(R7) ; buffer in NAM block
08 A7 01 90 00D1 281 MOVAB #NAM$R_PWD,NAM$B_NOP(R7); Do not mask out password in node spec
00D5 282
00D5 283 :
00D5 284 : Link the secondary NAM to the secondary FAB.
00D5 285 :
00D5 286
00D5 287 MOVAL FAL$L_NAM2(R8),- ; Store NAM2 pointer in FAB2
28 AA 00D9 288 FAB$L_NAM(R10) ;
5A 0200 C8 DE 00DB 289 MOVAL FAL$L_FAB(R8),R10 ; Restore R10 to FAB pointer
50 01 D0 00E0 290 MOVL #1,R0 ; Return success code
05 00E3 291 RSB
```



```
00F9 312      .SBTTL SUPPORT ROUTINES
000000F9 313      .PSECT FAL$CODE      NOSHR,EXE,RD,NOWRT,BYTE
00F9 314
00F9 315      :++
00F9 316      : Functional Description:
00F9 317      :
00F9 318      : The following routines are called by other action routines, not by the
00F9 319      : state table manager (FAL$STATE).
00F9 320      :
00F9 321      : These routines initialize the specified XAB. For Allocation and Key
00F9 322      : Definition XABs, the area ID and key of reference value, respectively,
00F9 323      : is an input parameter.
00F9 324      :
00F9 325      : Calling Sequence:
00F9 326      :
00F9 327      :     BSBW      FAL$INIT_ALLXAB
00F9 328      :     BSBW      FAL$INIT_DATXAB
00F9 329      :     BSBW      FAL$INIT_KEYXAB
00F9 330      :     BSBW      FAL$INIT_PROXAB
00F9 331      :     BSBW      FAL$INIT_RDTXAB
00F9 332      :     BSBW      FAL$INIT_SUMXAB
00F9 333      :
00F9 334      : Input Parameters:
00F9 335      :
00F9 336      :     R6      AID value for FAL$INIT_ALLXAB; REF value for FAL$INIT_KEYXAB
00F9 337      :
00F9 338      : Implicit Inputs:
00F9 339      :
00F9 340      :     None
00F9 341      :
00F9 342      : Output Parameters:
00F9 343      :
00F9 344      :     R0      Status code
00F9 345      :     R1-R5   Destroyed
00F9 346      :     R6      Unchanged
00F9 347      :     R7      Address of XAB initialized
00F9 348      :
00F9 349      : Implicit Outputs:
00F9 350      :
00F9 351      :     None
00F9 352      :
00F9 353      : Completion Codes:
00F9 354      :
00F9 355      :     R0      1 = success; 0 = failure
00F9 356      :
00F9 357      : Side Effects:
00F9 358      :
00F9 359      :     None
00F9 360      :
00F9 361      :--
00F9 362      :
00F9 363      :     ASSUME XAB$B_COD+1 EQ XAB$B_BLN
00F9 364      :     ASSUME FAL$K_ALLXAB GE XAB$C_ALLLEN
00F9 365      :     ASSUME FAL$K_DATXAB GE XAB$C_DATLEN
00F9 366      :     ASSUME FAL$K_KEYXAB GE XAB$C_KEYLEN_V2
00F9 367      :     ASSUME FAL$K_PROXAB GE XAB$C_PROLEN
00F9 368      :     ASSUME FAL$K_RDTXAB GE XAB$C_RDTLEN
```

FALACTINI
V04-000

- STATE TABLE ACTION ROUTINES
SUPPORT ROUTINES

I 3

16-SEP-1984 01:33:36
5-SEP-1984 01:16:05

VAX/VMS Macro V04-00
[FAL.SRC]FALACTINI.MAR;1

Page 9
(7)

00F9 369

ASSUME FAL\$K_SUMXAB GE XAB\$C_SUMLN

FAL
V04


```
00F9 371 .SBTTL FALSINIT_DATXAB, FALSINIT_PROXAB
00F9 372 .SBTTL FALSINIT_SUMXAB, FALSINIT_RDTXAB
00F9 373 .SBTTL FALSINIT_ALLXAB, FALSINIT_KEYXAB
00F9 374
00F9 375 :++
00F9 376 : This routine initializes the Date and Time XAB.
00F9 377 :--
00F9 378
57 0320 C8 DE 00F9 379 FALSINIT_DATXAB:: : Entry point
00F9 380 MOVAL FALS_DATXAB(R8),R7 : Get address of DATXAB
00FE 381 $ZERO_FILL- : Zero DATXAB
00FE 382 DST=(R7)- :
00FE 383 SIZE=#FALS DATXAB :
2C12 8F B0 0104 384 MOVW #<XAB$C_DAT[EN@8+XAB$C_DAT]>,- :
67 0108 385 XAB$B COD(R7) : Insert DATXAB ID and length
0091 31 0109 386 BRW INIT_SUC : All done
010C 387
010C 388 :++
010C 389 : This routine initializes the Protection XAB.
010C 390 :--
010C 391
57 034C C8 DE 010C 392 FALSINIT_PROXAB:: : Entry point
010C 393 MOVAL FALS_PROXAB(R8),R7 : Get address of PROXAB
0111 394 $ZERO_FILL- : Zero PROXAB
0111 395 DST=(R7)- :
0111 396 SIZE=#FALS PROXAB :
5813 8F B0 0119 397 MOVW #<XAB$C_PRO[EN@8+XAB$C_PRO]>,- :
67 011D 398 XAB$B COD(R7) : Insert PROXAB ID and length
7D 11 011E 399 BRB INIT_SUC : All done
0120 400
0120 401 :++
0120 402 : This routine initializes the Summary XAB.
0120 403 :--
0120 404
57 03A4 C8 DE 0120 405 FALSINIT_SUMXAB:: : Entry point
0120 406 MOVAL FALS_SUMXAB(R8),R7 : Get address of SUMXAB
0125 407 $ZERO_FILL- : Zero SUMXAB
0125 408 DST=(R7)- :
0125 409 SIZE=#FALS SUMXAB :
0C16 8F B0 012B 410 MOVW #<XAB$C_SUM[EN@8+XAB$C_SUM]>,- :
67 012F 411 XAB$B COD(R7) : Insert SUMXAB ID and length
6B 11 0130 412 BRB INIT_SUC : All done
0132 413
0132 414 :++
0132 415 : This routine initializes the Revision Date and Time XAB.
0132 416 :--
0132 417
57 03B0 C8 DE 0132 418 FALSINIT_RDTXAB:: : Entry point
0132 419 MOVAL FALS_RDTXAB(R8),R7 : Get address of RDTXAB
0137 420 $ZERO_FILL- : Zero RDTXAB
0137 421 DST=(R7)- :
0137 422 SIZE=#FALS RDTXAB :
141E 8F B0 013D 423 MOVW #<XAB$C_RDT[EN@8+XAB$C_RDT]>,- :
67 0141 424 XAB$B COD(R7) : Insert RDTXAB ID and length
59 11 0142 425 BRB INIT_SUC : All done
0144 426
0144 427 :++
```

```
0144 428 ; This routine initializes the Allocation XAB (by area ID).
0144 429 ;--
0144 430
0144 431 FALSINIT_ALLXAB:: ; Entry point
1F 56 D1 0144 432 CMPL R6,#FAL$K_MAX_AID ; Return error if area ID value
58 1A 0147 433 BGTRU INIT_ERR ; is too large
50 20 56 C5 0149 434 MULL3 R6,#FAL$K_ALLXAB,R0 ; Using AID as an index, compute
57 0C00 C840 9E 014D 435 MOVAB FAL$L_ALLXAB(R8)[R0],R7 ; address of Allocation XAB to use
0153 436 $ZERO_FILL- ; Zero ALLXAB
0153 437 DST=(R7)-
0153 438 SIZE=#FAL$K_ALLXAB
2014 8F B0 0159 439 MOVW #<XAB$C_ALL[EN@8+XAB$C_ALL],-
67 015D 440 XAB$B_COD(R7) ; Insert ALLXAB ID and length
17 A7 56 90 015E 441 MOVAB R6,XAB$B_AID(R7) ; Store area ID value
0162 442 $SETBIT R6,FAL$L_ALLXABINI(R8) ; Denote which ALLXAB was initialized
34 11 0167 443 BRB INIT_SUC ; All done
0169 444
0169 445 ;++
0169 446 ; This routine initializes the Key Definition XAB (by key of reference).
0169 447 ;--
0169 448
0169 449 FALSINIT_KEYXAB:: ; Entry point
1F 56 D1 0169 450 CMPL R6,#FAL$K_MAX_REF ; Return error if key of reference value
33 1A 016C 451 BGTRU INIT_ERR ; is too large
50 0000004C 8F 56 C5 016E 452 MULL3 R6,#FAL$K_KEYXAB,R0 ; Using REF as an index, compute
57 1000 C840 9E 0176 453 MOVAB FAL$L_KEYXAB(R8)[R0],R7 ; address of KEYXAB to use
017C 454 $ZERO_FILL- ; Zero KEYXAB
017C 455 DST=(R7)-
017C 456 SIZE=#FAL$K_KEYXAB
50 20 56 C5 0184 457 MULL3 R6,#FAL$K_KEYNAM,R0 ; Using REF as an index, compute
38 A7 1C00 C840 9E 0188 458 MOVAB FAL$L_KEYNAM(R8)[R0],- ; address of key name buffer to use
018F 459 XAB$B_KNM(R7) ; and store address in XAB
4015 8F B0 018F 460 MOVW #<XAB$C_KEYLEN_V2@8+XAB$C_KEY>,-
67 0193 461 XAB$B_COD(R7) ; Insert KEYXAB ID and length
17 A7 56 90 0194 462 MOVAB R6,XAB$B_REF(R7) ; Store key of reference value
0198 463 $SETBIT R6,FAL$L_KEYXABINI(R8) ; Denote which KEYXAB was initialized
019D 464
019D 465 ;++
019D 466 ; Common exit paths.
019D 467 ;--
019D 468
019D 469 INIT_SUC: ;
50 01 D0 019D 470 MOVL #1,R0 ; Return success code
05 01A0 471 RSB ; Exit
01A1 472 INIT_ERR: ;
50 D4 01A1 473 CLRL R0 ; Return failure code
05 01A3 474 RSB ; Exit
01A4 475
01A4 476 .END ; End of module
```


FALACTINI
Symbol table

- STATE TABLE ACTION ROUTINES

L 3

16-SEP-1984 01:33:36 VAX/VMS Macro V04-00
5-SEP-1984 01:16:05 [FAL.SRC]FALACTINI.MAR;1

Page 12
(8)

DAP\$B_BITCNT 00000035
DAP\$B_BLKCNT 00000056
DAP\$B_CTLFUNC 00000040
DAP\$B_DCODE_FID 00000019
DAP\$B_DCODE_MAC 0000001B
DAP\$B_DCODE_MSG 0000001A
DAP\$B_FLAGS 00000031
DAP\$B_KRF 00000047
DAP\$B_LEN256 00000034
DAP\$B_LENGTH 00000033
DAP\$B_RAC 00000046
DAP\$B_STREAMID 00000032
DAP\$B_TYPE 00000030
DAP\$B_X_FIELD 00000024
DAP\$C_BCN 000000C0
DAP\$K_BLN 000000C0
DAP\$K_RAC_D = 00000000
DAP\$K_SEQ_ACC = 00000000
DAP\$K_VALID_R2F = 0000EDBE
DAP\$L_CMWA 00000030
DAP\$L_CRC_RSLT 00000020
DAP\$L_DCODE_STS 00000018
DAP\$L_MSG_MASK 0000001C
DAP\$L_ROP 00000050
DAP\$L_SSPWA 00000080
DAP\$L_TEMP 00000090
DAP\$M_BITCNT = 00000008
DAP\$M_BLKCNT = 00000040
DAP\$M_SEGMENT = 00000040
DAP\$M_TMP1\$ = 00000008
DAP\$M_TMP2\$ = FFF80000
DAP\$Q_DCODE_FLG 00000000
DAP\$Q_KEY 00000048
DAP\$Q_MSG_BUF1 00000008
DAP\$Q_MSG_BUF2 00000010
DAP\$Q_SYSPEC 00000038
DAP\$W_CTLMENU 00000044
DAP\$W_DISPLAY2 00000054
DAP\$W_PARTNER 00000006
DAP\$W_VERSION 00000004
FAB\$B_BID = 00000000
FAB\$B_BLN = 00000001
FAB\$C_BID = 00000003
FAB\$C_BLN = 00000050
FAB\$L_FNA = 0000002C
FAB\$L_FOP = 00000004
FAB\$L_NAM = 00000028
FAB\$L_XAB = 00000024
FAB\$V_NAM = 00000018
FAL\$B_ACCFUNC 000001F6
FAL\$B_ACCOPT 000001F5
FAL\$B_DATATYPE 000001F4
FAL\$B_DISABLE 00000006
FAL\$B_ENABLE 00000005
FAL\$B_LOGGING 00000004
FAL\$B_MISCOPT 00000007
FAL\$B_RAC 000001F7

FAL\$B_RBK_CACHE 00000012
FAL\$B_RCVBUFIDX 00000011
FAL\$B_VALUE 00000010
FAL\$C_WRKBLN 00002000
FAL\$INIT 00000000 RG 02
FAL\$INIT_ALLXAB 00000144 RG 02
FAL\$INIT_DATXAB 000000F9 RG 02
FAL\$INIT_KEYXAB 00000169 RG 02
FAL\$INIT_PROXAB 0000010C RG 02
FAL\$INIT_RDTXAB 00000132 RG 02
FAL\$INIT_RENAME 00000094 RG 02
FAL\$INIT_SUMXAB 00000120 RG 02
FAL\$INIT_XABCHN 000000E4 RG 02
FAL\$K_ALCXAB = 00000020
FAL\$K_DATXAB = 0000002C
FAL\$K_EXPAND = 000000FF
FAL\$K_EXPAND2 = 000000FF
FAL\$K_FAB = 00000050
FAL\$K_FAB2 = 00000050
FAL\$K_FHCXAB = 0000002C
FAL\$K_KEYNAM = 00000020
FAL\$K_KEYXAB = 0000004C
FAL\$K_MAX_AID = 0000001F
FAL\$K_MAX_REF = 0000001F
FAL\$K_NAM = 00000060
FAL\$K_NAM2 = 00000060
FAL\$K_PROXAB = 00000058
FAL\$K_RAB = 00000044
FAL\$K_RDTXAB = 00000014
FAL\$K_RESULT = 000000FF
FAL\$K_RESULT2 = 000000FF
FAL\$K_SUMXAB = 0000000C
FAL\$K_WRKBLN 00002000
FAL\$L_ALLXAB 00000C00
FAL\$L_ALLXABINI 00000074
FAL\$L_CHAIN_NXT 0000007C
FAL\$L_DATXAB 00000320
FAL\$L_FAB 00000200
FAL\$L_FAB2 00000800
FAL\$L_FHCXAB 000002F4
FAL\$L_FOP 000001F8
FAL\$L_KEYNAM 00001C00
FAL\$L_KEYXAB 00001000
FAL\$L_KEYXABINI 00000078
FAL\$L_NAM 00000294
FAL\$L_NAM2 00000850
FAL\$L_NUMBER 000001FC
FAL\$L_PROXAB 0000034C
FAL\$L_RAB 00000250
FAL\$L_RCVBUF 0000005C
FAL\$L_RDTXAB 000003B0
FAL\$L_RMS_PTR 0000006C
FAL\$L_STB 000000C0
FAL\$L_SUMXAB 000003A4
FAL\$L_TEMP 000003F4
FAL\$L_USE_SC1 000000A8
FAL\$L_USE_SC2 000000AC

FAL
V04

FALACTINI
Symbol table

- STATE TABLE ACTION ROUTINES

M 3

16-SEP-1984 01:33:36 VAX/VMS Macro V04-00
5-SEP-1984 01:16:05 [FAL.SRC]FALACTINI.MAR;1

Page 13
(8)

FAL\$L_USE_VER	= 000000A4		
FAL\$M_ATT_MSG	= 00000002		
FAL\$M_BLK_IO	= 00000200		
FAL\$M_FTM	= 00000100		
FAL\$M_WILD	= 00000400		
FAL\$Q_BLD	00000050		
FAL\$Q_DIRNAME	00000088		
FAL\$Q_FALLOG	00000090		
FAL\$Q_FLG	00000000		
FAL\$Q_MBX	00000038		
FAL\$Q_MBXIOSB	00000030		
FAL\$Q_RCV	00000040		
FAL\$Q_RCVIOSB	00000020		
FAL\$Q_RMS	00000064		
FAL\$Q_STATE_CTX	00000008		
FAL\$Q_SYSNET	00000098		
FAL\$Q_TEMP	000003F8		
FAL\$Q_VOLNAME	00000080		
FAL\$Q_XMT	00000048		
FAL\$Q_XMTIOSB	00000028		
FAL\$T_DAP	00000100		
FAL\$T_DIRNAME	00001F00		
FAL\$T_EXPAND	00000500		
FAL\$T_EXPAND2	00000A00		
FAL\$T_FALLOG	00001C00		
FAL\$T_FILESPEC	00000400		
FAL\$T_FILESPEC2	00000900		
FAL\$T_KEYBUF	00000700		
FAL\$T_MBXBUF	00001980		
FAL\$T_PRTBUF1	00001A00		
FAL\$T_PRTBUF2	00001800		
FAL\$T_RESULT	00000600		
FAL\$T_RESULT2	00000B00		
FAL\$T_SYSNET	00001D00		
FAL\$T_VOLNAME	00001E00		
FAL\$V_CNF_MSG	= 00000000		
FAL\$W_DAPBUFSIZ	0000001A		
FAL\$W_DISPLAY	00000070		
FAL\$W_LNKCHN	0000001C		
FAL\$W_MBXCHN	0000001E		
FAL\$W_QIOBUFSIZ	00000018		
FAL\$W_RECEIVED	00000072		
FAL\$W_USE_DBS	000000A0		
FAL\$W_USE_SYS	000000A2		
INIT_ERR	000001A1	R	02
INIT_SUC	0000019D	R	02
NAM\$B_BID	= 00000000		
NAM\$B_BLN	= 00000001		
NAM\$B_ESS	= 0000000A		
NAM\$B_NOP	= 00000008		
NAM\$B_RSS	= 00000002		
NAM\$C_BID	= 00000002		
NAM\$C_BLN	= 00000060		
NAM\$L_ESA	= 0000000C		
NAM\$M_RSA	= 00000004		
NAM\$M_PWD	= 00000001		
RAB\$B_BID	= 00000000		

RAB\$B_BLN	= 00000001
RAB\$C_BID	= 00000001
RAB\$C_BLN	= 00000044
RAB\$C_FAB	= 0000003C
XAB\$B_AID	= 00000017
XAB\$B_BLN	= 00000001
XAB\$B_COD	= 00000000
XAB\$B_REF	= 00000017
XAB\$C_ALL	= 00000014
XAB\$C_ALLLEN	= 00000020
XAB\$C_DAT	= 00000012
XAB\$C_DATLEN	= 0000002C
XAB\$C_FHC	= 0000001D
XAB\$C_FHCLEN	= 0000002C
XAB\$C_KEY	= 00000015
XAB\$C_KEYLEN_V2	= 00000040
XAB\$C_PRO	= 00000013
XAB\$C_PROLEN	= 00000058
XAB\$C_RDT	= 0000001E
XAB\$C_RDTLEN	= 00000014
XAB\$C_SUM	= 00000016
XAB\$C_SUMLLEN	= 0000000C
XAB\$C_KNM	= 00000038
XAB\$C_NXT	= 00000004

FAL
V04

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes
ABS	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABSS	00002000 (8192.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
FAL\$CODE	000001A4 (420.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.07	00:00:01.13
Command processing	146	00:00:00.41	00:00:02.11
Pass 1	322	00:00:08.11	00:00:28.29
Symbol table sort	5	00:00:01.04	00:00:03.28
Pass 2	94	00:00:01.56	00:00:10.03
Symbol table output	25	00:00:00.13	00:00:00.81
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	625	00:00:11.35	00:00:45.68

The working set limit was 1650 pages.

64571 bytes (127 pages) of virtual memory were used to buffer the intermediate code.

There were 60 pages of symbol table space allocated to hold 1130 non-local and 4 local symbols.

476 source lines were read in Pass 1, producing 13 object records in Pass 2.

25 pages of virtual memory were used to define 24 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
_\$255\$DUA28:[FAL.OBJ]FAL.MLB;1	6
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	15
TOTALS (all libraries)	21

1320 GETS were required to define 21 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:FALACTINI/OBJ=OBJ\$:FALACTINI MSRC\$:FALACTINI/UPDATE=(ENH\$:FALACTINI)+LIB\$:FAL/LIB

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY